EMBEDDED SYSTEMS LAB

MINI-PROJECT

INTERFACING A TEMPERATURE SENSOR WITH LP1768 MICROCONTROLLER

TEAM DETAILS:

Batch: A1

Team No: 4

Branch: CCE

Team Member Details:

SNO	**NAME**	**REG-NO**
1	Chakka Venkata Sai Manikanta Santhan	190953022
2	Sankalp Srivastava	190953027
3	Ritika Singh Kartik	190953029

PROBLEM STATEMENT

Write a program to interface a temperature sensor to LPC1768 and display the temperature on LCD.

Temperature is one of the basic properties of nature. Knowing the temperature of an object or the environment in useful and needed in several situations be it Daily Life or Industrial Applications. Over the years there have been several types of temperature sensors Thermocouples, RTDs, Thermistors, Infrared, and Semiconductor Sensors.

In this project we have interfaced a LM35 Temperature Sensor with a LPC 1768 ARM Microcontroller.

The aim of this project is to:-

1)To comprehend the software development for ARM cortex microcontroller using embedded C language.

2)To design real world systems using ARM cortex-M embedded system.

3)To understand various interfacing circuits necessary for various applications and programming using ARM.

INTRODUCTION

The following components have been used

LPC1768

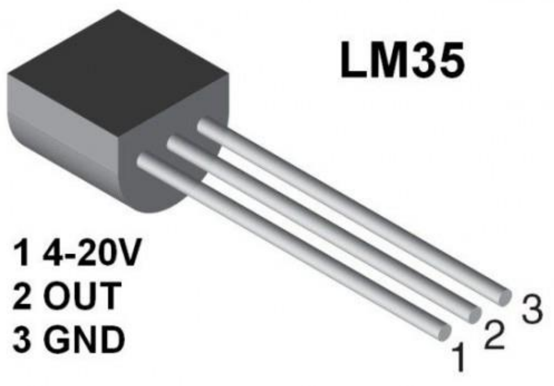
The NXP (founded by Philips) LPC1768 is an ARM 32-bit CortexM3 Microcontroller. The code has been written in Embedded C and tested out on a ALS evaluation board.

A 12-bit internal Analog-to-Digital Converter has been used to convert the voltage value to its corresponding value in degree Celsius.

LM35 Temperature Sensor

//The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature. (Linear at 10.0 mV/°C scale factor)//

LM35 is a well known temperature sensor and is directly calibrated in Degrees Celsius meaning that the output voltage is directly proportional to Degrees Celsius readings. It has a measurement range is between -55°C to 150°C giving typical accuracy(s) of 0.25°C at room temperature and 0.75°C for full range.



FEATURES

• Calibrated directly in **Celsius**

(Centigrade)
• Linear **+ 10-mV/°C** Scale Factor
• **0.5°C** Ensured Accuracy (at 25°C)
• Rated for Full **–55°C** to **150°C** Range • Suitable for Remote Applications
• Operates From **4 V** to **30 V**
• Less than 60-μA Current Drain
• Low Self-Heating, 0.08°C in Still Air

• Non-Linearity Only ±1⁄4°C Typical

• Low-Impedance Output, 0.1 Ω for 1-mA Load

HARDWARE COMPONENTS USED

Components Name	**Quantity**
Bread board	**1**
ALS-SDA-ARMCTXM3-01	1
Power Supply (+5V)	1
Cross Cable	1
LM35 Temperature Sensor	1
Jumper Cables (Female-Male)	4
Hair Dryer	1
USB port in the computer and PC for downloading the software	1

SOFTWARE REQUIREMENTS:

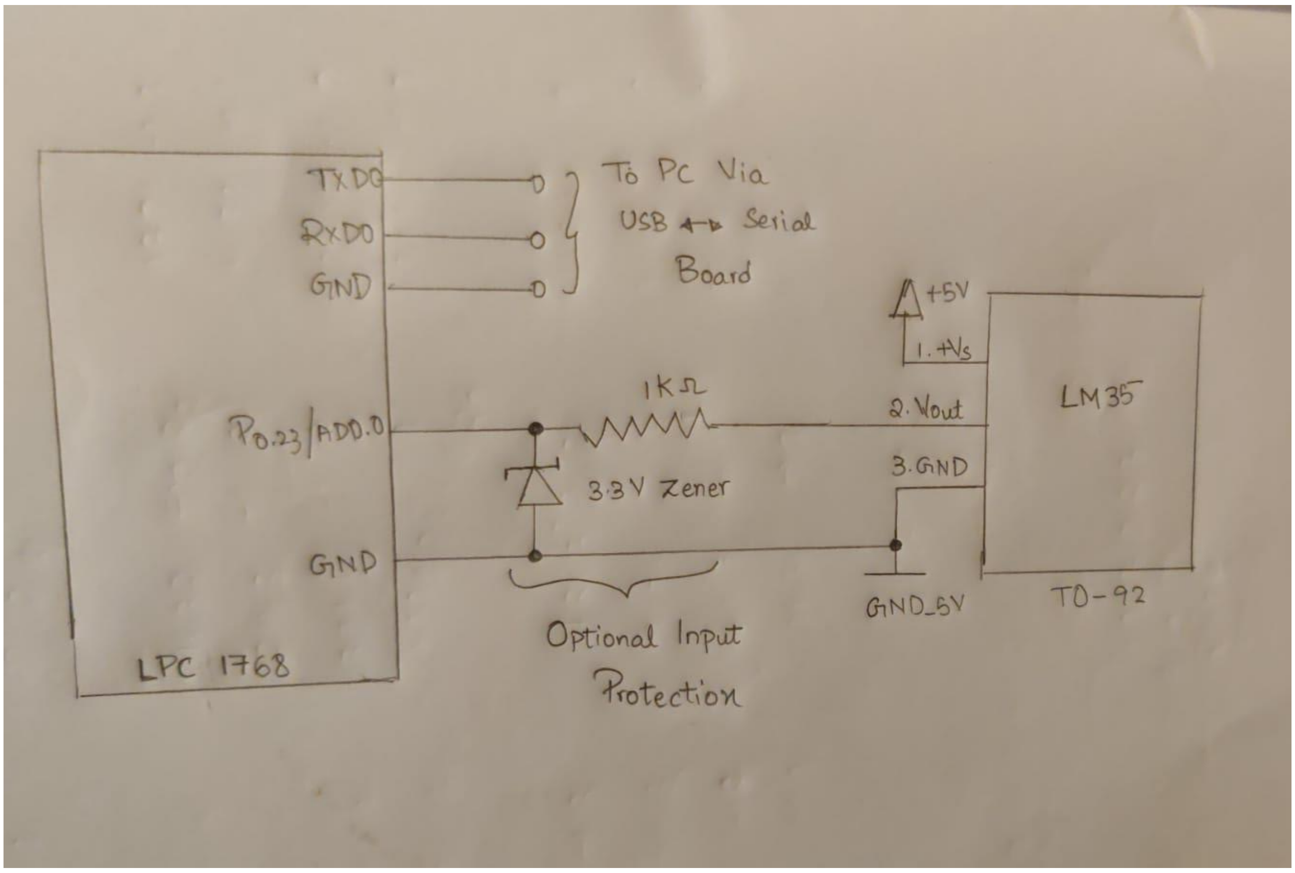
• Programming Language: Embedded C

• KEIL UVision IDE

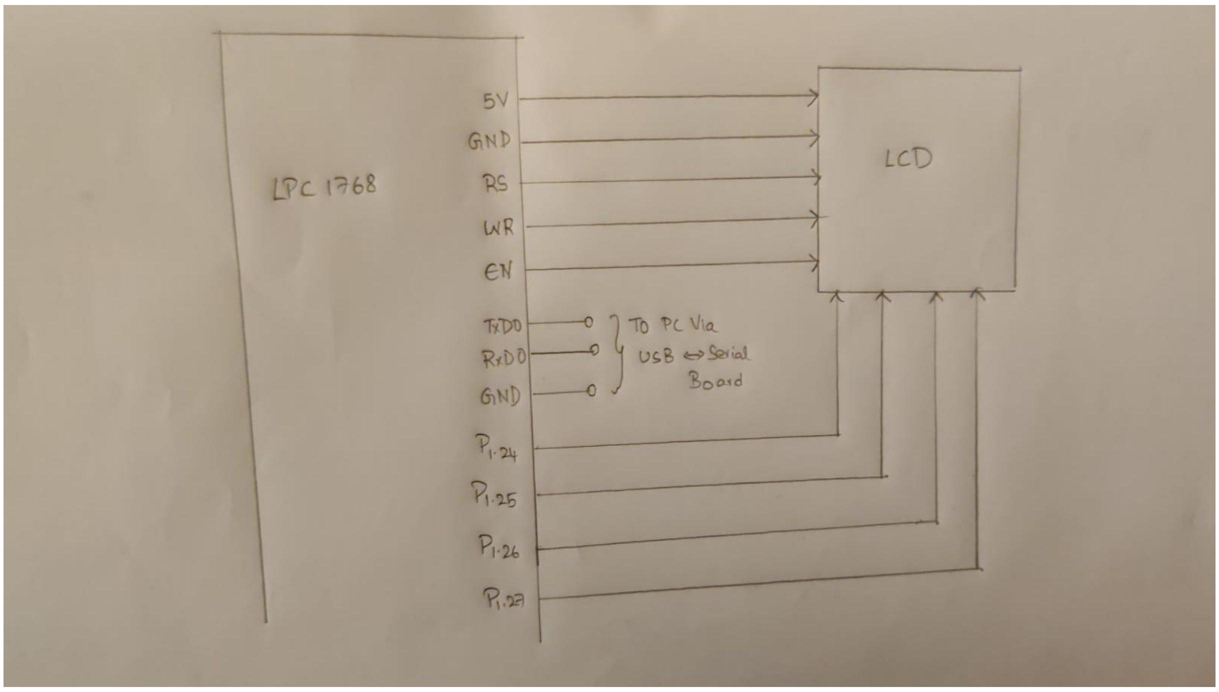
• Flash Magic

PIN DIAGRAMS

LM35 temperature sensor connection to LPC1768:



Connection to LCD:



FLOWCHART AND EXPLANATION

The following steps show how to get temperature from the sensor reading:

Step 1-->Configure pins, port pin 0.1-0.7 (configuring lcd 4bit mode), p0.8 as Rs pin and p0.9 as enable pin.

Step 2-->Configure ADC Registers

Step 3-->Load addr0 register value into temp

Step 4-->Extract value by loading temp with 0xfff0 and right shift by 4

Step 5-->In final (a float variable) store the value of temperature after dividing it by 12.41

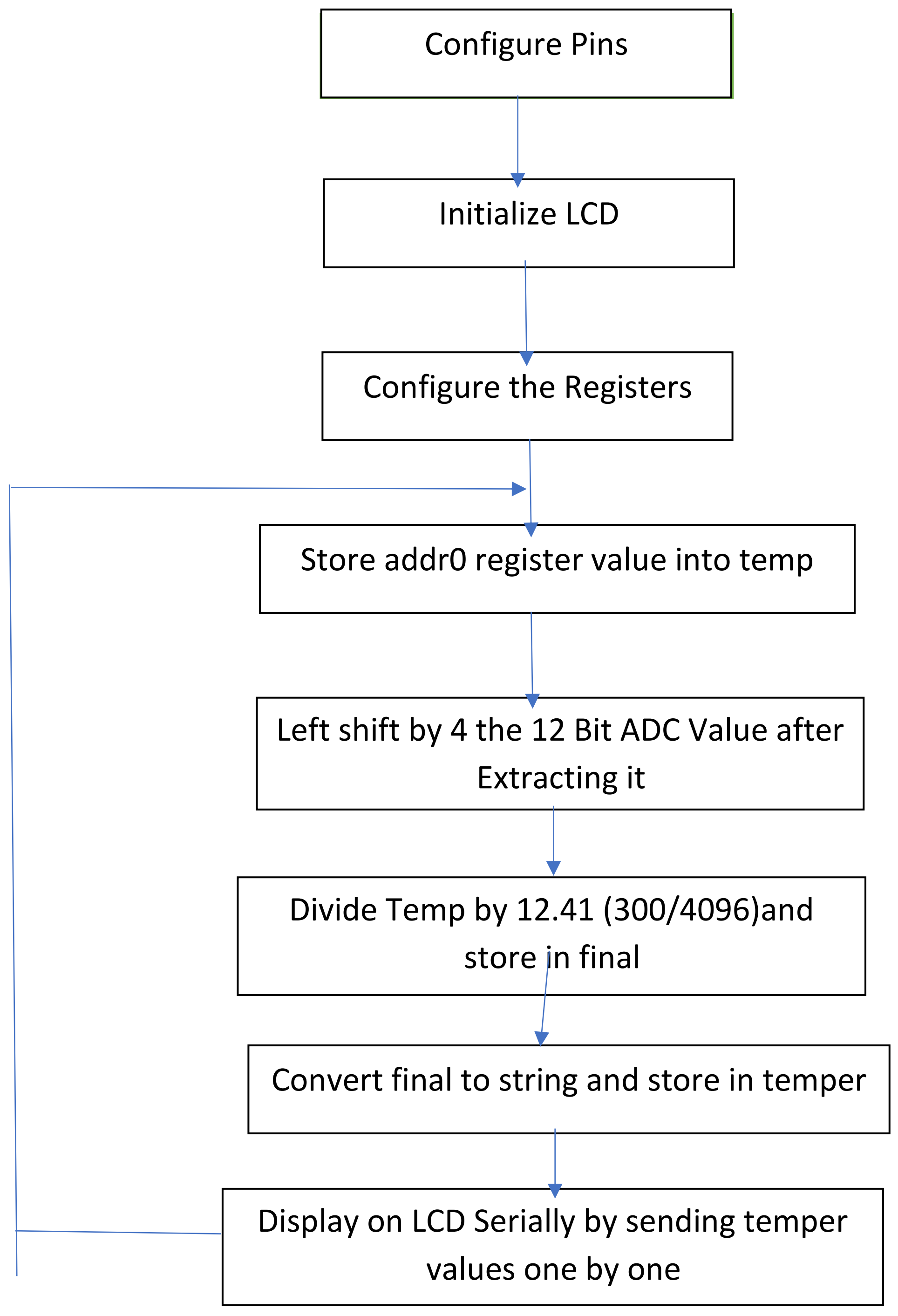
Step 6-->Convert the final value to string and store into variable temper

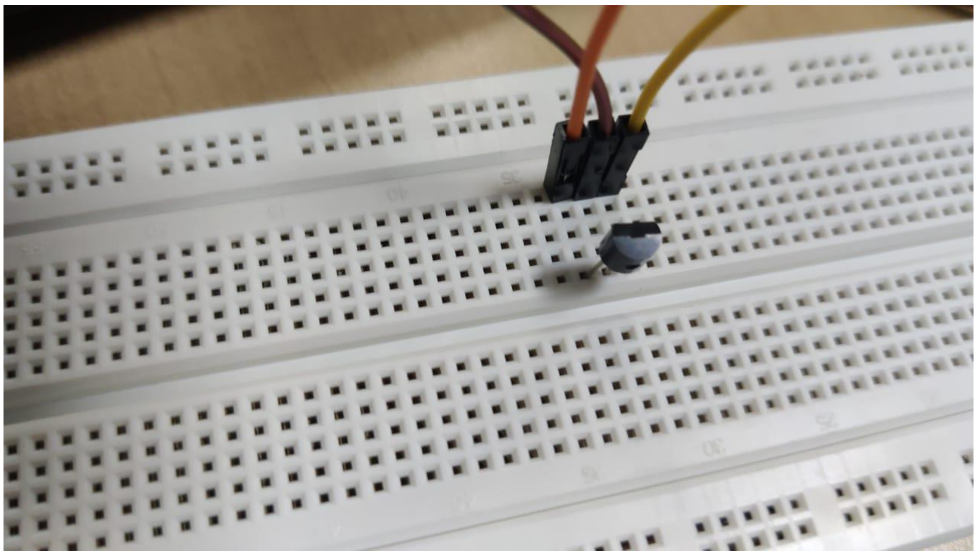
Step 7-->Display String on LCD serially

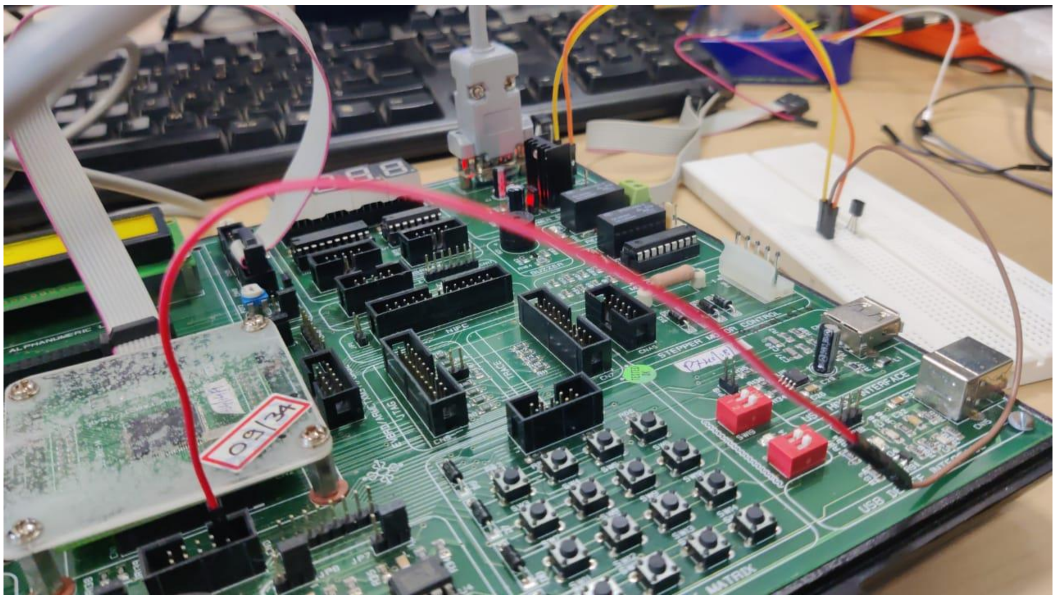
Step Size of ADC=3.3 / (2^12) = 0.0806mV.

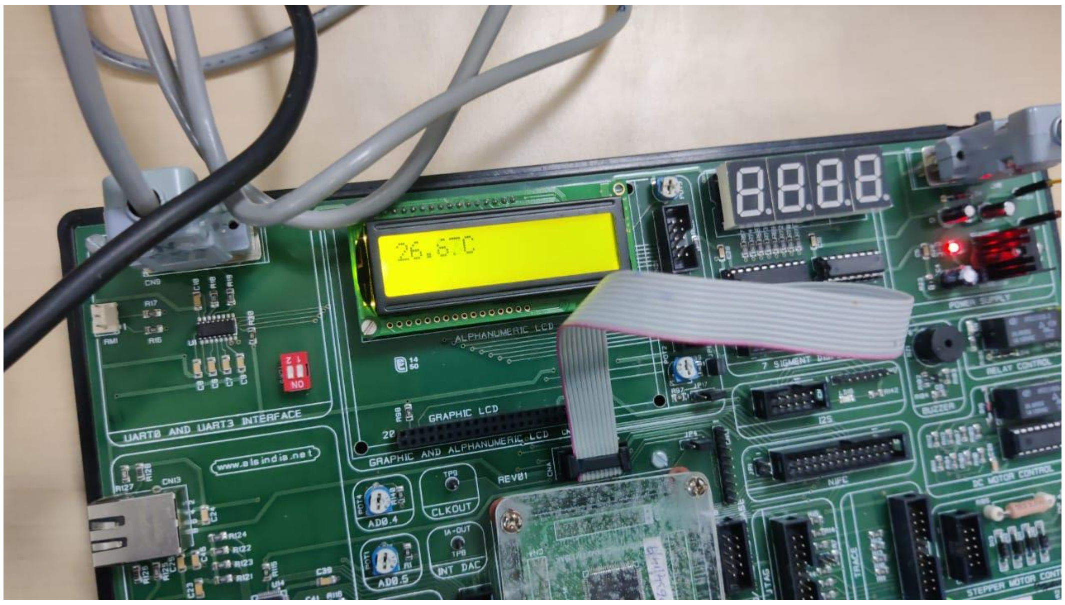
For Every Degree C Lm35 provides 10mV of change.

Hence ADC Value is divided by 12.41 to get Temperature


Configure Pins
Initialize LCD
Configure the Registers
Store addr0 register value into temp
Left shift by 4 the 12 Bit ADC Value after Extracting it
Divide Temp by 12.41 (300/4096)and store in final
Convert final to string and store in temper
Display on LCD Serially by sending temper values one by one






26.67C
ALPHANUMERIC LCD
GRAPHIC LCD
UART0 AND UART3 INTERFACE
www.alsindia.net
STEPPER MOTOR CONTROL
DC MOTOR CONTROL
RELAY CONTROL
BUZZER
POWER SUPPLY

CODE

```
 #include<LPC17xx.h>
#include <stdio.h>
unsigned int i,j, temp;
float final;
char temper[20];
unsigned long LED =0x00000010;
#define rsctrl 0x00000100 //po.8
#define enctrl 0x00000200 //po.9 enable lcd
#define dtctrl 0x000000f0 //p0.4-7 4 bit mode
#define VREF       3.3
void lcd_init();
void wr_cn();
void clr_disp();
void delay();
void lcd_com();
void wr_dn();
void lcd_data();
void clear_ports();
void lcd_puts(unsigned char *);
unsigned int i,temp1=0,temp2=0;
int main(void)
{
SystemInit() ;
SystemCoreClockUpdate();
LPC_PINCON->PINSEL1|=(1<<14);
while(1)
{
LPC_ADC->ADCR=(1<<0) | (1<<21) | (1<<24);
while(((temp=LPC_ADC->ADDR0) & (1<<31))==0);
temp=LPC_ADC->ADDR0;
temp&=0xFFF0;
temp>>=4;
final=((float)temp * VREF * 100)/4096;
sprintf(temper,"%3.2fC",final);
lcd_init();
temp1=0x80;
lcd_com();
delay(10000);
lcd_puts(&temper[0]);
}
}
void lcd_init(){
LPC_PINCON->PINSEL0=0;
LPC_GPIO0->FIODIR|=dtctrl;
LPC_GPIO0->FIODIR|=rsctrl;
LPC_GPIO0->FIODIR|=enctrl;
clear_ports();
delay(3200);
for(i=0;i<3;i++)
{
temp2=(0x30);
wr_cn();
delay(30000);
}
temp2=(0x20);
wr_cn(); delay(30000);
temp1=0x28; lcd_com();
delay(30000); temp1=0x0c;
lcd_com(); delay(800);
temp1=0x06; lcd_com();
delay(800); temp1=0x01;
lcd_com(); delay(10000);
temp1=0x80; lcd_com();
delay(800);
}
void lcd_com()
{ temp2=temp1&0xf0;
temp2=temp2;
wr_cn();
temp2=temp1&0x0f;
temp2=temp2<<4;
wr_cn(); delay(1000);
}
void wr_cn()
{
clear_ports();
LPC_GPIO0->FIOPIN=temp2;
LPC_GPIO0->FIOCLR=rsctrl;
LPC_GPIO0->FIOSET=enctrl;
delay(25);
LPC_GPIO0->FIOCLR=enctrl;
}
void lcd_data()
{
temp2=temp1&0xf0;
temp2=temp2;
wr_dn();
temp2=temp1&0x0f;
temp2=temp2<<4;
wr_dn();
delay(1000);
}
void wr_dn()
{
clear_ports();
LPC_GPIO0->FIOPIN=temp2;
LPC_GPIO0->FIOSET=rsctrl;
LPC_GPIO0->FIOSET=enctrl;
delay(25);
LPC_GPIO0->FIOCLR=enctrl;
}
void delay(unsigned int r1)
{
unsigned int r;
for(r=0;r<r1;r++);
}
void clr_disp()
{
temp1=0x01;
lcd_com();
delay(10000);
}
void clear_ports()
```

```
{
LPC_GPIO0->FIOCLR|=rsctrl;
LPC_GPIO0->FIOCLR|=enctrl;
LPC_GPIO0->FIOCLR|=dtctrl;
}
void lcd_puts(unsigned char *buff)
{
unsigned int i=0;
while(buff[i]!='\0')
{
temp1=buff[i]; i++; lcd_data();
}
}
```

RESULT

We used a hair dryer to observe the change in the temperature

Minimum Temperature: 24.4 C

Maximum Temperature : 75 C

OBSERVATION

Initially the temperature on the LCD was displayed as room temperature. Upon using the hair dryer on the sensor, we could observe a gradual rise in temperature. After we turned it off, there was a gradual fall back to the room temperature corresponding to the sensor being back to its normal room temperature.